🏁 Teongbin 포팅 매뉴얼

목차

🔨 기술 스택

▼ Frontend

- Node.js - 20.11.0
- Vite - 5.3.1
- Vue.js - vue@3.4.33
- Bootstrap - 5.3.3

▼ Backend

- JAVA - Oracle Open JDK 17
- SpringBoot - 3.3.2

▼ Infra Structure

- ubuntu - 20.04 (LTS)

    - nginx - 1.18.0
    - docker - 27.1.1
    - Jenkins - 2.471
- ▼ Databse
    - Redis - 5.0.7
    - MariaDB - 10.3.39

💡 환경 변수 설정

1. Frontend

```
# vite 환경변수  ( .env )
    VITE_APP_MAP_API_KEY='{key}'
```

2. Backend

```
# application-prod.properties

# DB settings
spring.application.name=teongbin
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

# JPA settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect
spring.sql.init.mode=always
spring.datasource.url=jdbc:mariadb://i11c101.p.ssafy.io:3310/
spring.datasource.username=
spring.datasource.password=
spring.resources.static-locations=classpath:/static/
spring.data.redis.host=
spring.data.redis.port=
spring.data.redis.password=
spring.mail.username=
spring.mail.password=

# JWT settings
jwt.SECRET=C101
jwt.EXPIRATION_TIME=8640000
jwt.TOKEN_PREFIX=Bearer
jwt.HEADER_STRING=Authorization

# Timezone
spring.jackson.time-zone=Asia/Seoul
spring.jpa.properties.hibernate.jdbc.time_zone=Asia/Seoul
```

✈️ 배포 방법

1. Nginx 설치 및 설정

- Nginx 설치

```
sudo apt-get install nginx
```

- Nginx 설정

```
server {
    if ($host = i11c101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name i11c101.p.ssafy.io;

}
```

```
server {
        listen 443 ssl;
        server_name i11c101.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/i11c101.p.ssafy
        ssl_certificate_key /etc/letsencrypt/live/i11c101.p.s

        # Other SSL options
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # Main Page
        location /main/ {
#                    proxy_pass http://172.17.0.5:80/;
            proxy_pass http://localhost:8083/;
             proxy_set_header X-Real-IP $remote_addr;
             proxy_set_header X-Forwarded-For $proxy_add_x_fo
             proxy_set_header Host $host;
                         proxy_set_header X-Forwarded-Proto $
        }

        # Sub Page
        location / {
            proxy_pass http://localhost:8082/;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        # Jenkins Page
        location /ssafy/jenkins/ {
                proxy_pass http://localhost:8081/ssafy/jenkin
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_redirect http://localhost:8081/ssafy/je
        }

        # Backend Page
        location /api/v1 {
                proxy_pass http://localhost:8080/api/v1;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x
                proxy_set_header X-Forwarded-Proto $scheme;
                }
        }
```

- HTTPS 인증 (Certbot)

```
sudo snap install certbot --classic
sudo certbot certonly --standalone -d 도메인이름
```

2. Docker 설치 및 Dockerfile 구성

- Docker's apt repository

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

- Install latest Docker version

```
sudo apt-get install docker-ce docker-ce-cli containerd.io do
```

1. Frontend Dockerfile + nginx.conf

   - Dockerfile

```
# Dockerfile for Vue.js Project
# Stage 1: Build the application
FROM node:latest AS build-stage

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

# Stage 2: Serve the application
FROM nginx:alpine AS production-stage

COPY --from=build-stage /app/dist /usr/share/nginx/html

COPY ./nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

   - nginx.conf

```
server {
    listen 80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

2. Backend Dockerfile

```
FROM openjdk:17
ARG JAR_FILE=build/libs/teongbin-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=prod",
```

3. Docker Compose

```
services:
  backend:
    build:
      context: ./backend
    image: ssafy/teongbin
    container_name: teongbin
    ports:
      - "8080:8080"
    env_file:
      - .env.backend  # Spring Boot용 환경 파일

  dashboard:
    build:
      context: ./frontend/dashboard
    image: ssafy/dashboard
    container_name: dashboard
    ports:
      - "8083:80"
  product_page:
    build:
      context: ./frontend/product-page
    image: ssafy/product_page
    container_name: product_page
    ports:
      - "8082:80"
```

3. Jenkins 설치 및 설정

- Jenkins 설치 및 환경 변수 설정

```
# Jenkins 설치

cd /home/ubuntu && mkdir jenkins-data

sudo ufw allow *8080*/tcp
sudo ufw reload
sudo ufw status

sudo docker run -d -p 8080:8080 --env JENKINS_OPTS="--prefix=
-v /home/ubuntu/.ssh:/root/.ssh -v /var/run/docker.sock:/var/

sudo docker logs jenkins

sudo docker stop jenkins
sudo docker ps -a

# 환경 설정 변경

cd /home/ubuntu/jenkins-data

mkdir update-center-rootCAs

wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center

sudo sed -i 's#https://updates.jenkins.io/update-center.json#
sudo docker restart jenkin
```

- Jenkins 초기 비밀번호 설정 후, 접속
- Gitlab과 Webhook 설정
- Jenkins 내부 Docker 설치

```
:~$ docker exec -it jenkins-container bash
:/# apt-get update
:/# apt-get install vim
:/# apt-get install wget
:/# apt install apt-transport-https ca-certificates curl soft
:/# wget -qO- https://get.docker.com/ | sh
:/# apt-get install docker-ce docker-ce-cli containerd.io doc
```

- Jenkins Scripts 설정

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'ssafy/teongbin'
        CONTAINER_NAME = 'teongbin'
        GIT_CREDENTIALS_ID = 'a22fdc86-8de3-410f-b584-bf36dbb
        GIT_URL = 'https://lab.ssafy.com/s11-webmobile3-sub2/
        GIT_BRANCH = 'master'

        VUE_MAIN_IMAGE = 'ssafy/dashboard'
        VUE_SUB_IMAGE = 'ssafy/product_page'
        VUE_MAIN_CON = 'dashboard'
        VUE_SUB_CON = 'product_page'
    }

    stages {
        stage('GitLab에서 코드 가져오기') {
            steps {
                git branch: "${GIT_BRANCH}",
                    credentialsId: "${GIT_CREDENTIALS_ID}",
                    url: "${GIT_URL}"
            }
        }

        stage('Secret 파일 로드') {
            steps {
                script {
                    withCredentials([file(credentialsId: 'fro
                                     file(credentialsId: 'bac
                    sh '''
                        echo "Workspace directory: $WORKSPACE
                        ls -al $WORKSPACE
                        cp $FRONT_ENV_FILE $WORKSPACE/.env
                        cp $BACKEND_ENV_FILE $WORKSPACE/.env.
                        echo "Frontend env file copied to: $W
                        echo "Backend env file copied to: $WO
                    '''
                    }
                }
            }
        }

        stage('기존 컨테이너 제거') {
            steps {
                script {
                    sh 'docker rm -f teongbin || true'
                    sh 'docker rm -f $VUE_MAIN_CON|| true'
                    sh 'docker rm -f $VUE_SUB_CON|| true'
                    sh 'docker rmi ssafy/teongbin || true'
                    sh 'docker rmi $VUE_MAIN_IMAGE || true'
                    sh 'docker rmi $VUE_SUB_IMAGE || true'
                }
            }
        }

        stage('Docker Compose 빌드 및 실행') {
            steps {
                script {
                    try {
                        // Docker Compose에 환경 파일 전달
                        sh 'docker compose down || true'
                        sh 'docker compose build --no-cache'
                        sh 'docker compose up -d'
                    } catch (e) {
                        error "Docker Compose 빌드 또는 실행 실패
                    }
                }
            }
        }
    }

    post {
        success {
            script {
                def commitMessage = sh(script: "git log -1 --
                mattermostSend (
                    color: 'good',
                    message: ":jenkins7: 배포 성공    :wow_taxi2
                )
            }
        }
        failure {
            script {
                mattermostSend (
                    color: 'danger',
                    message: ":jenkins7: 배포 실패    :j_dragon_
                )
            }
        }
    }
}
```